BACCALAURÉAT GÉNÉRAL BLANC

NSI

Epreuve de spécialité

DURÉE DE L'ÉPREUVE : **3 h 30**

L'usage d'une calculatrice N´EST PAS autorisé

Exercice 1: **ROUTAGE** (5 POINTS)
Exercice 2: **FESTIVAL DE JAZZ** (5 POINTS)
Exercice 3: **AU SUPERMARCHÉ** (5 POINTS)
Exercice 4: **RÉCURSIVITÉ** (5 POINTS)

EXERCICE 1: ROUTAGE (5 points)

Cet exercice porte sur les représentations binaires et les protocoles de routage.

1. Une adresse IPv4 est représentée sous la forme de 4 nombres séparés par des points. Chacun de ces 4 nombres peut être représenté sur un octet.

a. Donner en écriture décimale l'adresse IPv4 correspondant à l'écriture binaire : 11000000.10101000.10000000.10000011

b. Tous les ordinateurs du réseau A ont une adresse IPv4 de la forme : 192.168.128._ _ _ , où seul le dernier octet (représenté par _ _ _) diffère. Donner le nombre d'adresses machines différentes possibles du réseau A.

2. On rappelle que le protocole RIP cherche à minimiser le nombre de routeurs traversés (qui correspond à la métrique). On donne les tables de routage d'un réseau informatique composé de 5 routeurs (appelés A, B, C, D et E), chacun associé directement à un réseau du même nom obtenues avec le protocole RIP :

Routeur A

Destination	Métrique
A	0
B	1
C	1
D	1
E	2

Routeur B

Destination	Métrique
A	1
B	0
C	2
D	1
E	2

Routeur C

Destination	Métrique
A	1
B	2
C	0
D	1
E	2

Routeur D

Destination	Métrique
A	1
B	1
C	1
D	0
E	1

Routeur E

Destination	Métrique
A	2
B	2
C	2
D	1
E	0

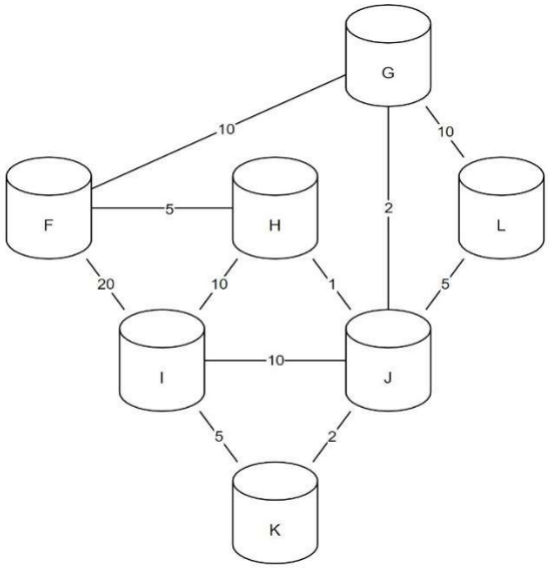
a. Donner la liste des routeurs avec lesquels le routeur A est directement relié.

b. Représenter graphiquement et de manière sommaire les 5 routeurs ainsi que les liaisons existantes entre ceux-ci.

3. Le protocole OSPF est un protocole de routage qui cherche à minimiser la somme des métriques des liaisons entre routeurs.
Dans le protocole de routage OSPF le débit des liaisons entre routeurs agit sur la métrique via la relation : $métrique = \frac{10^8}{débit}$ dans laquelle le débit est exprimé en bit par seconde (bps).
On rappelle qu'un kbps est égal à 10^3 bps et qu'un Mbps est égal à 10^6 bps.
Recopier sur votre copie et compléter le tableau suivant :

Débit	100 kbps	500 kbps	?	100 Mbps
Métrique associée	1 000	?	10	1

4. Voici la représentation d'un réseau et la table de routage incomplète du routeur F obtenue avec le protocole OSPF :



Routeur F

Destination	Métrique
F	0
G	8
H	5
I	
J	
K	
L	

Les nombres présents sur les liaisons représentent les coûts des routes avec le protocole OSPF.

a. Indiquer le chemin emprunté par un message d'un ordinateur du réseau F à destination d'un ordinateur du réseau I. Justifier votre réponse.

b. Recopier et compléter la table de routage du routeur F.

c. Citer une unique panne qui suffirait à ce que toutes les données des échanges de tout autre réseau à destination du réseau F transitent par le routeur G. Expliquer en détail votre réponse.

EXERCICE 2: FESTIVAL DE JAZZ (5 points)

On pourra utiliser les mots clés SQL suivants : `AND`, `FROM`, `INSERT`, `INTO`, `JOIN`, `OR`, `ON`, `SELECT`, `SET`, `UPDATE`, `VALUES`, `WHERE`.

On étudie une base de données permettant la gestion de l'organisation d'un festival de musique de jazz, dont voici le schéma relationnel comportant trois relations :

- la relation `groupes` (`idgrp`, `nom`, `style`, `nb_pers`)
- la relation `musiciens` (`idmus`, `nom`, `prenom`, `instru`, `#idgrp`)
- la relation `concerts` (`idconc`, `scene`, `heure_debut`, `heure_fin`, `#idgrp`)

Dans ce schéma relationnel :

- les clés primaires sont soulignées ;
- les clés étrangères sont précédées d'un #.

Ainsi `concerts.idgrp` est une clé étrangère faisant référence à `groupes.idgrp`.

Voici un extrait des tables `groupes`, `musiciens` et `concerts` :

extrait de `groupes`

idgrp	nom	style	nb_pers
12	'Weather Report'	'Latin Jazz'	5
25	'Breckers Brothers'	'Swing Jazz'	4
87	'Return to Forever'	'Latin Jazz'	8
96	'The Jazz Messenger'	'Free Jazz'	3

extrait de `musiciens`

idmus	nom	prenom	instru	idgrp
12	'Parker'	'Charlie'	'trompette'	96
13	'Parker'	'Charlie'	'trombone'	25
58	'Dufler'	'Candy'	'saxophone'	96
97	'Miles'	'Davis'	'saxophone'	87

extrait de `concerts`

idconc	scene	heure_debut	heure_fin	idgrp
10	1	'20h00'	'20h45'	12
24	2	'20h00'	'20h45'	15
36	1	'21h00'	'22h00'	96
45	3	'18h00'	'18h30'	87

Figure 1 : Extrait des tables `groupes`, `musiciens` et `concerts`

1. Citer les attributs de la table `groupes`.
2. Justifier que l'attribut `nom` de la table `musiciens` ne peut pas être une clé primaire.
3. En s'appuyant uniquement sur l'extrait des tables fourni dans la *figure 1* écrire ce que renvoie la requête :

```
SELECT nom
FROM groupes
WHERE style = 'Latin Jazz';
```

4. Le concert dont l'`idconc` est 36 finira à 22h30 au lieu de 22h00. Recopier sur la copie et compléter la requête SQL ci-dessous permettant de mettre à jour la relation `concerts` pour modifier l'horaire de fin de ce concert.

```
UPDATE concerts
SET …
WHERE … ;
```

5. Donner une requête SQL permettant de récupérer le nom de tous les groupes qui jouent sur la *scène 1*.

6. Fournir une requête SQL permettant d'ajouter dans la relation `groupes` le groupe 'Smooth Jazz Fourplay', de style 'Free Jazz', composé de 4 membres. Ce groupe aura un `idgrp` de 15.

Les données sont ensuite récupérées pour être analysées par la société qui produit les festivals de musique. Pour ce faire, elle utilise la programmation en Python afin d'effectuer certaines opérations plus complexes.

Elle stocke les données relatives aux musiciens sous forme d'un *tableau de dictionnaires* dans laquelle a été ajouté le nombre de concerts effectués par chaque musicien :

```
>>> print(musiciens)
  [{'idmus': 12, 'nom': 'Parker', 'prenom': 'Charlie',
  'instru': 'trompette', 'idgrp' : 96, 'nb_concerts': 5},
  {'idmus': 13, 'nom': 'Parker', 'prenom': 'Charlie',
  'instru': 'trombone', 'idgrp' : 25, 'nb_concerts': 9},
  {'idmus': 58, 'nom': 'Dufler', 'prenom': 'Candy',
  'instru': 'saxophone', 'idgrp' : 96, 'nb_concerts': 4},
  {'idmus': 97, 'nom': 'Miles', 'prenom': 'Davis',
  'instru': 'saxophone', 'idgrp' : 87, 'nb_concerts': 2},
  …
  ]
```

7. Écrire la fonction `recherche_nom` ayant pour unique paramètre un tableau de dictionnaires (comme `musiciens` présenté précédemment) renvoyant un tableau contenant le nom de tous les musiciens ayant participé à au moins 4 concerts.

EXERCICE 3: AU SUPERMARCHÉ (5 points)

Cet exercice porte sur les structures de données (files et la programmation objet en langage python)

Un supermarché met en place un système de passage automatique en caisse.
Un client scanne les articles à l'aide d'un scanner de code-barres au fur et à mesure qu'il les ajoute dans son panier.
Les articles s'enregistrent alors dans une structure de données.

La structure de données utilisée est une file définie par la classe `Panier`, avec les primitives habituelles sur la structure de file.
Pour faciliter la lecture, le code de la classe `Panier` n'est pas écrit.

```
class Panier():
  def __init__(self):
     """Initialise la file comme une file vide."""

  def est_vide(self):
     """Renvoie True si la file est vide, False sinon."""

  def enfiler(self, e):
     """Ajoute l'élément e en dernière position de la
    file, ne renvoie rien."""


  def defiler(self):
     """Retire le premier élément de la file et le renvoie."""
```

Le panier d'un client sera représenté par une file contenant les articles scannés.
Les articles sont représentés par des tuples (`code_barre, designation, prix, horaire_scan`) où:

- `code_barre` est un nombre entier identifiant l'article ;
- `designation` est une chaine de caractères qui pourra être affichée sur le ticket de caisse ;
- `prix` est un nombre décimal donnant le prix d'une unité de cet article ;
- `horaire_scan` est un nombre entier de secondes permettant de connaitre l'heure où l'article a été scanné.

1. On souhaite ajouter un article dont le tuple est le suivant (`31002, "café noir", 1.50, 50525`).
   Ecrire le code utilisant une des quatre méthodes ci-dessus permettant d'ajouter l'article à l'objet de classe `Panier` appelé `panier1`.
2. On souhaite définir une méthode `remplir(panier_temp)` dans la classe `Panier` permettant de remplir la file avec tout le contenu d'un autre panier `panier_temp` qui est un objet de type `Panier`.

Recopier et compléter le code de la méthode `remplir` en remplaçant chaque `.....` par la primitive de file qui convient.

```
def remplir(self, panier_temp):
    while not panier_temp.....:
        article = panier_temp.....
        self.....(article)
```

3. Pour que le client puisse connaitre à tout moment le montant de son panier, on souhaite ajouter une méthode `prix_total()` à la classe `Panier` qui renvoie la somme des prix de tous les articles présents dans le panier.
   Ecrire le code de la méthode `prix_total`. **Attention, après l'appel de cette méthode, le panier devra toujours contenir ses articles**.

4. Le magasin souhaite connaitre pour chaque client la durée des achats. Cette durée sera obtenue en faisant la différence entre le champ `horaire_scan` du dernier article scanné et le champ `horaire_scan` du premier article scanné dans le panier du client.

   Un panier vide renverra une durée égale à zéro. On pourra accepter que le panier soit vide après l'appel de cette méthode.

   Ecrire une méthode `duree_courses` de la classe `Panier` qui renvoie cette durée.

EXERCICE 4: RÉCURSIVITÉ (5 points)

Cet exercice est consacré à l'analyse et à l'écriture de programmes récursifs.

1.

a. Expliquer en quelques mots ce qu'est une fonction récursive.

b. On considère la fonction Python suivante :

Numéro de lignes	Fonction `compte_rebours`
1	`def compte_rebours(n):`
2	`    """ n est un entier positif ou nul """`
3	`    if n >= 0:`
4	`        print(n)`
5	`        compte_rebours(n - 1)`

L'appel `compte_rebours(3)` affiche successivement les nombres 3, 2, 1 et 0. Expliquer pourquoi le programme s'arrête après l'affichage du nombre 0.

2. En mathématiques, la factorielle d'un entier naturel *n* est le produit des nombres entiers strictement positifs inférieurs ou égaux à *n*. Par convention, la factorielle de 0 est 1. Par exemple :

la factorielle de 1 est 1
la factorielle de 2 est 2 x 1 =2
la factorielle de 3 est 3 x 2 x 1 = 6
la factorielle de 4 est 4 x 3 x 2 x 1 = 24…

Recopier et compléter sur votre copie le programme donné ci-dessous afin que la fonction récursive `fact` renvoie la factorielle de l'entier passé en paramètre de cette fonction.

Exemple : `fact(4)` renvoie 24.

Numéro de lignes	Fonction `fact`
1	`def fact(n):`
2	`    """ Renvoie le produit des nombres entiers`
3	`    strictement positifs inférieurs à n """`
4	`    if n == 0:`
5	`        return` ***`à compléter`***
6	`    else:`
7	`        return` ***`à compléter`***

3. La fonction `somme_entiers_rec` ci-dessous permet de calculer la somme des entiers, de `0` à l'entier naturel `n` passé en paramètre.

Par exemple :

Pour `n` = 0, la fonction renvoie la valeur 0.

Pour `n` = 1, la fonction renvoie la valeur 0 + 1 = 1.

…

Pour `n` = 4, la fonction renvoie la valeur 0 + 1 + 2 + 3 + 4 = 10.

Numéro de lignes	Fonction `somme_entiers_rec`

1
2
3
4
5
6
7
8
	`def somme_entiers_rec(n):`
	`    """ Permet de calculer la somme des entiers,`
	`    de 0 à l'entier naturel n """`
	`    if n == 0:`
	`        return 0`
	`    else:`
	`        print(n) #pour vérification`
	`        return n + somme_entiers_rec(n - 1)`

L'instruction `print(n)` de la ligne 7 dans le code précédent a été insérée afin de mettre en évidence le mécanisme en œuvre au niveau des appels récursifs.

a. Écrire ce qui sera affiché dans la console après l'exécution de la ligne suivante : `res = somme_entiers_rec(3)`

b. Quelle valeur sera alors affectée à la variable `res` ?

4. Écrire en Python une fonction `somme_entiers` non récursive : cette fonction devra prendre en argument un entier naturel `n` et renvoyer la somme des entiers de 0 à `n` compris. Elle devra donc renvoyer le même résultat que la fonction `somme_entiers_rec` définie à la question *3*.

Exemple : `somme_entiers(4)` renvoie 10.